Empirical Software Engineering

CSE 8340 — Fall 2002

Prof. Jeff Tian, tian@engr.smu.edu
CSE, SMU, Dallas, TX 75275
(214) 768-2861; Fax: (214) 768-3085
www.engr.smu.edu/~tian/class/8340.02f

Module IVb: Tian-Zelkowitz Study

- A Theory of Program Complexity;

- A Technique for Measure Evaluation;

- Application, Validation and Perspectives.
General Idea/Strategy

• A *theory* of program complexity:
  ▶ a formal theory/model/framework
  ▶ organizes empirical knowledge
  ▶ systematically extends earlier studies by Prather, Fenton and Whitty, and Weyuker.

• An evaluation/selection *technique*:
  ▶ basis: theory above
  ▶ procedure: constrained optimization
  ▶ extrapolate measurement activities to new applications

• *Validate* the model using data from NASA Software Engineering Laboratory.
Tian-Zelkowitz Theory

- **Axioms:** Define program complexity and state common properties.

- **Dimensionality Analysis:**
  provide the basis for metrics classification
  - Aspects or dimensions:
    presentation, control, data;
  - Levels: lexical, syntactic, semantic.

- **Classification Scheme:** Define mutually exclusive and collectively exhaustive classes.
Theory: Axiom Overview

**Complexity:** Relationship between program-pairs;

**Comparability:** Programs with identical functionality are comparable (A1); Composite programs are comparable to their components (A2).

**Monotonicity:** Sufficiently large programs will become more complex (A3).

**Measurability:** Measures on programs must agree with underlying complexity (A4).

**Diversity:** Distribution of measured complexity must not form a single cluster (A5).
Theory: Defining Complexity

Definition: A complexity ranking $\mathcal{R}$ is a binary relation on programs. Given programs $P$ and $Q$, we interpret $\mathcal{R}(P,Q)$ as $P$ being no more complex than $Q$.

\[ C(P,Q) \text{ iff } \mathcal{R}(P,Q) \lor \mathcal{R}(Q,P). \]

Comments:

- It is *internal* to the programs;
- Related empirical to external properties;
- Very broad definition, need further qualification and quantification.
Theory: Comparability Axioms

Axiom A1: \( (\forall P, Q) \ ( \square P = \square Q \Rightarrow C(P, Q) ) \)

i.e., functionally equivalent programs are comparable.

Axiom A2: \( (\forall P, Q) \ (IN(P, Q) \Rightarrow C(P, Q) ) \)

i.e., a composite program is comparable with its components.

Comments:

- Hard problem due to undecidability;
- \( \mathcal{R} \) is self-reflexive;
- \( \mathcal{R} \) is not transitive.
Theory: Monotonicity Axiom

- **Axiom A3**: \( (\exists K \in \mathbb{N})(\forall P, Q) \)
  \( (IN(P, Q) \land (dist(P, Q) > K)) \Rightarrow R(P, Q) \) 
  i.e., sufficiently large programs will not be ranked lower in complexity.

- **Comments:**
  - General trend must be followed;
  - Local deviations allowed.
Theory: Measure Definition Axiom

**Definition:** A *complexity measure* \( \mathcal{V} \) is a quantification of complexity ranking \( \mathcal{R} \). \( \mathcal{V} \) maps programs into real numbers:

\[
\mathcal{V} : \mathbb{U} \Rightarrow \mathcal{R}
\]

**Axiom A4:** \( (\forall P, Q) \ (\mathcal{R}(P, Q) \Rightarrow \mathcal{V}(P) \leq \mathcal{V}(Q)) \)
i.e., a measure must agree with the ranking it is approximating.

**Non-Axiom:** Commonly assumed by other complexity models:

\[
(\forall P, Q) \ (\mathcal{V}(P) \leq \mathcal{V}(Q) \Rightarrow \mathcal{R}(P, Q))
\]

▷ Incomparable cases;
▷ Non-transitive cases;
Theory: Distribution Axiom

Requirement: Measure values must not cluster around one single dominating point.

Axiom A5: \((\forall k \in \mathbb{R})(\exists \delta > 0)\)
\[
(|U - \{P : \mathcal{V}(P) \in [k - \delta, k + \delta]\}| = |U|)
\]

Axiom A5': \((\forall k \in \mathbb{R})(\exists \delta > 0)\)
\[
\sum_{P, \mathcal{V}(P) \in [k-\delta, k+\delta]} prob(P) < 1
\]

Rationale: A single dominating cluster is disallowed because it fails to achieve the goal of providing comparison for programs.
Theory: Complexity Dimensions

Presentation: Physical presentation for readers that has no effect on functionality.

Control: Instructions, control structures, and control dependencies.

Data: Data items, data structures, and data dependencies.

Comments:

▷ Control + Data = Abstract;
▷ Orthogonal dimensions.
Theory: Measurement Levels

**Lexical:** Token based measure computation;

**Syntactic:** Directly syntax based measure computation;

**Semantic:** Semantic analysis needed for measure computation.

Comments:

▷ 27 possible points in a 3-D space;
▷ Space proximity \( \approx \) Measure similarity;
▷ Dividing control and data dimensions: 1. count, 2. structure, 3. dependency.
**Theory: Vertical Classification**

Classification based on computational models used:

- Depend only on syntax trees of programs?
  Yes, Abstract. No, non-abstract.
- Invariant to renaming?
  Yes, Functional. No, non-functional.

\[
\begin{align*}
\text{Abstract} & \quad \text{Functional} \\
\text{Non Abstract} & \quad \text{Non Functional}
\end{align*}
\]
Theory: Vertical Classification Example

```
\[
\begin{array}{ll}
\text{Abstract} & \{ \text{scan, stmt, ss, fp, cyc, knot, du, hac, ac} \} \\
\text{Non Abstract} & \{ lc \} \\
\text{Functional} & \{ \text{stmt, ss, fp, cyc, knot, du, hac, ac} \} \\
\text{Non Functional} & \{ \text{scan} \}
\end{array}
\]
```
Theory: Hierarchical Classification

Classification based on complexity relations of component-composite programs:

- Sensitive to context?
  - Yes, interactional. No, context free.
- Depend only on building element but not organization?
  - Yes, Primitive. No, non-primitive.
- Capture both interface and internal?
  - Yes, Overall. No, non-overall.

All

- Context Free
  - Primitive
  - Non Primitive
- Interactional
  - Overall
  - Non Overall
Theory: Hier. Classification Example

All

Context Free
\{ scan, stmt, ss, cyc, knot \}

Interactional
\{ fp, du, hac, ar \}

Primitive
\{ scan, stmt, ss, cyc, knot \}

Non Primitive
\{ \}

Overall
\{ du, hac, ar \}

Non Overall
\{ fp \}
Evaluation: Problems and Solutions

Problem: Evaluation of complexity measures;

Assumption: Measures satisfy Axiom A4;

View: Measures as points in a measure space;

Solution Strategy:

- Define feasible region by using axioms and classification as boundary conditions;
- Derive scales for measures within the feasible region;
- Aggregate evaluations and select the optimal measure.
Evaluation: Boundary Conditions

Axioms as testable predicates:

**BC**₁. Axiom A1: \( (\mathcal{D}(\mathcal{V}) — \text{domain of } \mathcal{V} ) \)
\[
(\forall P, Q)( \left[ \begin{array}{c} \mathbb{P} \end{array} \right] = \left[ \begin{array}{c} \mathbb{Q} \end{array} \right] \wedge P \in \mathcal{D}(\mathcal{V}) ) \Rightarrow Q \in \mathcal{D}(\mathcal{V})
\]

**BC**₂. Axiom A2:
\[
(\forall P, Q)( (\text{IN}(P, Q) \lor \text{IN}(Q, P)) \wedge P \in \mathcal{D}(\mathcal{V} ) )
\]

**BC**₃. Modified Axiom A3:
\[
(\exists K)(\forall P, Q)( \text{dist}(P, Q) > K ) \Rightarrow \mathcal{V}(P) \leq \mathcal{V}(Q)
\]

**BC**₄. Assumed true.

**BC**₅. Modified Axiom A5:
\[
(\forall k \in \mathbb{R})(\exists \delta > 0) \text{prob}(\mathcal{V}(P) \in [k-\delta, k+\delta]) < 1
\]
Evaluation: Screening Using Axioms

Example Measure: $\mathcal{V}(P) = 1 - \frac{1}{s(P)}$, where $s(P)$ is the statement count of $P$.

Screening:

- $\mathbf{BC}_1$ is satisfied because $\mathcal{D}(\mathcal{V}) = \mathcal{U}$;
- $\mathbf{BC}_2$ same as above;
- $\mathbf{BC}_3$ is satisfied because:
  \[(\forall P, Q) \ (0 < s(P) < s(Q)) \Rightarrow \left( 1 - \frac{1}{s(P)} < 1 - \frac{1}{s(Q)} \right)\]
- $\mathbf{BC}_5$ is not satisfied because:
  \[prob(\mathcal{V}(P) \in [1 - \delta, 1 + \delta]) = 1\]

Result: Reject $\mathcal{V}$ due to violation of $\mathbf{BC}_5$. 

Evaluation: Screening Using Classes

BC₆: Appropriate class depends on goals.

Goal 1. Documentation vs. comprehension.
   Target: Non-abstract class.
   Reject: Abstract class.

Goal 2. Object code size assessment.
   Target: Abstract class.
   ▶ Total line count might be acceptable;
   ▶ Blank line count is rejected.

Goal 3. Programming effort prediction.
   Target: Both abstract & non-abstract classes.
   Reason: Both contribute to effort.
Evaluation: Monotonicity Scale

Assumption: Prefer measures that better approximates monotonicity;

Need to capture the extent and frequency of non-monotonic deviations;

Scale $S_1$: The monotonicity scale is $\langle T, p_m \rangle$, where $T$ is the period of monotonicity:

$$T = \min_K \left( \text{dist}(P, Q) > K \Rightarrow \forall(P) \leq \forall(Q) \right)$$

and $p_m$ is the conditional probability of non-monotonic component-composite pairs:

$$p_m = \text{prob}(\forall(P) > \forall(Q) \mid \text{IN}(P, Q))$$
Evaluation: Distribution Scale

Assumption: Uniform distribution desirable.

Need to capture:

▷ Significant points on the scale;
▷ Uniformity of these points.

Uniformity Scale $S_3$: For measure $\mathcal{V}$, $\epsilon > 0$, $\delta > 0$, and $p_k = prob(k\delta \leq \mathcal{V}(P) < (k+1)\delta)$, $S_3 = \langle n, \ d \rangle$, where $n$ and $d$ are the cardinality and the normalized s.d. of \{\(p_k\mid p_k > \epsilon\}\).

$$d = \begin{cases} 
0 & \text{if } n = 0 \\
\sqrt{\frac{\sum_k (1-np_k)^2}{n}} & \text{otherwise}
\end{cases}$$
Evaluation: Scale & Dominance Relation

Global Scaling Vector $g$ is defined on relevant scales $\{s_j\}$ with $g(\mathcal{V})[i]$ defined successively as:

$$g(\mathcal{V})[i] = \begin{cases} s_j(\mathcal{V})[k] & \text{if opt} = \max \\ -s_j(\mathcal{V})[k] & \text{if opt} = \min \end{cases}$$

until all individual scaling dimensions $s_j(\mathcal{V})[k]$ are exhausted.

Dominance Relation: A measure $\mathcal{V}_i$ is said to dominate another measure $\mathcal{V}_j$ if

$$\left( g(\mathcal{V}_i) \geq g(\mathcal{V}_j) \right) \land (\exists k)( g(\mathcal{V}_i)[k] > g(\mathcal{V}_j)[k] )$$

Elimination: All dominated measures are eliminated.
Evaluation: Objective Function

Assess the importance and trade-off among $S_j$ to form weight vector $W$.

$$(\forall i, j, k) \ W(V_i)[k] = W(V_j)[k] = W[k]$$

Example: the weight for $S_1$ could be $d \times p_c$.

The selection problem reduces to the constrained optimization problem:

$$\max_i \left( f_i = \sum_j G(V_i)[j] \times W[j] \right)$$

such that:

$V_i$ satisfies all boundary conditions.
Application and Model Validation

1. Application domain: risk identification for projects in NASA/SEL;

2. Pilot experiment: apply the scientific model to select complexity measures;

3. Data collection: run multiple applications and collect results;

4. Analyze resulting data-points to validate the scientific model.
Application: Risk Identification

Risk in software decisions:

- Multiple alternatives;
- Uncertainty about future development;
- Large investment;
- Significant consequences.

Risk Identification via CTA (Selby & Porter):

- Risk: likelihood of high cost or effort;
- High cost: highest quartile (80:20 rule);
- Basis: historical data;
- Methodology: classification trees.
Application: CTA Prediction Example

Predictions are made based on:

- Classification tree;
- Sample module measurement data:

<table>
<thead>
<tr>
<th>Modules</th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_4$</th>
<th>$m_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyclomatic complexity</td>
<td>3</td>
<td>8</td>
<td>13</td>
<td>30</td>
<td>45</td>
</tr>
<tr>
<td>function plus module call</td>
<td>8</td>
<td>40</td>
<td>7</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>operators count</td>
<td>30</td>
<td>18</td>
<td>10</td>
<td>33</td>
<td>58</td>
</tr>
<tr>
<td>module calls</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>prediction</td>
<td>–</td>
<td>?</td>
<td>–</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td>actual</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>+</td>
</tr>
</tbody>
</table>
Application: CTA Cost & Performance

Cost factors:

▶ Tree generation: measure pool size $S$;
▶ Tree usage: tree-complexity/node-count.

Performance Measures:

▶ Coverage: Predictions made;
▶ Accuracy: Correct predictions;
▶ Completeness: Correct predictions of actual high cost modules;
▶ Consistency: Correct high cost predictions.
Application: CTA Performance

Compare predicted and actual data:

Coverage \[= \frac{P}{P + N}\]

Accuracy \[= \frac{M_{11} + M_{22}}{P}\]

Completeness \[= \frac{M_{11}}{A_+}\]

Consistency \[= \frac{M_{11}}{P_+}\]

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>+/−</td>
<td>+/−</td>
</tr>
<tr>
<td>not identified</td>
<td>not identified</td>
</tr>
</tbody>
</table>
Pilot: Problem & Screening

Environment: NASA/SEL;

Goal: Identify high cost modules using complexity measures.
Consequence: Eliminate non-complexity measures, reducing \( S \) from 74 to 40.

Screening of measures:

- \( BC_1 \) and \( BC_2 \) true because \( D(Y) = U \);
- \( BC_3 \) eliminates right half of Table 2;
- \( BC_5 \) true from observing data;
- \( BC_6 \) true because all aspects contribute to total effort;
- Result: Measure pool \( S \) reduced from 40 to 18.
Pilot: Measure Selection

Criteria: Conformance between $\mathcal{V}$ and total-effort distribution. No need for $S_1$.

Derivation: Mark a quartile “+” if $p_i(\mathcal{V}) \geq 0.75$ and “−” if $n_i(\mathcal{V}) \geq 0.75$, where:

- $m_i(\mathcal{V}) = \#\text{modules in quartile } i$;
- $p_i(\mathcal{V}) = \text{prob}(m_4(\text{effort}) | m_i(\mathcal{V}))$
- $n_i(\mathcal{V}) = 1 - p_i(\mathcal{V})$.

$$\max_{\mathcal{V}, \mathcal{V} \in S} \left\{ \sum_{i=1}^{4} \left\{ m_i(\mathcal{V})p_i(\mathcal{V}) + m_i(\mathcal{V})n_i(\mathcal{V}) \right\} \right\}$$

- $p_i(\mathcal{V}) \geq 0.75$
- $\forall n_i(\mathcal{V}) \geq 0.75$
### Pilot: Prediction Result

<table>
<thead>
<tr>
<th>ACTA:</th>
<th>Actual</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>+</td>
<td>7</td>
<td>17</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>−</td>
<td>4</td>
<td>143</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>11</td>
<td>160</td>
<td>171</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OCTA:</th>
<th>Actual</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>+</td>
<td>7</td>
<td>32</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td></td>
<td>−</td>
<td>4</td>
<td>129</td>
<td>133</td>
<td></td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>11</td>
<td>161</td>
<td>172</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>performance measure</th>
<th>OCTA</th>
<th>ACTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>not identified</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>correctly identified</td>
<td>136</td>
<td>150</td>
</tr>
<tr>
<td>incorrectly identified</td>
<td>36</td>
<td>21</td>
</tr>
<tr>
<td>coverage</td>
<td>97%</td>
<td>97%</td>
</tr>
<tr>
<td>accuracy</td>
<td>79%</td>
<td>87%</td>
</tr>
<tr>
<td>completeness</td>
<td>63%</td>
<td>63%</td>
</tr>
<tr>
<td>consistency</td>
<td>17%</td>
<td>29%</td>
</tr>
</tbody>
</table>
Validation: Problems and Environment

**Goal:** Extrapolate pilot study result to validate our model.

**Embedded Environment:** NASA/SEL:

- 16 ground support projects;
- SLOC: 3K to 112K of Fortran code;
- Staffing: 4-23 (5-25M / 5-140 MM);
- Modules: 83-531/proj, 4700+ total;
- Measures: 74 collected.

**Direct Environment:** CTA:

- Training set size: 1;
- Testing on immediate next project;
- 10 data points from 16 raw data;
- 5 data points from isolated data.
Validation: Result Comparison

Overall Comparison:

<table>
<thead>
<tr>
<th>cost</th>
<th>measure pool</th>
<th>OCTA</th>
<th>ACTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree size (all)</td>
<td>12.5</td>
<td>9.1</td>
<td></td>
</tr>
<tr>
<td>tree size (-1)</td>
<td>7.3</td>
<td>4.4</td>
<td></td>
</tr>
</tbody>
</table>

| performance | coverage     | 97.6% | 97.0% |
| performance | accuracy     | 69.7% | 74.5% |
| performance | consistency  | 38.4% | 50.4% |
| performance | completeness | 35.6% | 36.0% |
Validation: Validation Result

1. Comparing with original CTA, measure selection using our model is effective:
   ▶ Cost: Measure pool size and classification tree complexity are reduced dramatically;
   ▶ Performance: Coverage and completeness remain virtually the same; Accuracy and consistency are improved.

2. Comparing with random guessing, CTA based on either measure selection method made great improvement, well worth the cost.

3. The multiple data-points indicate the validity of our model.
Validation: Baselines

Baseline 1: Original CTA.

Baseline 2: Optimal Random Guessing:
coverage = 100%
accuracy = 62.5%
completeness = 25%
consistency = 25%

Comment: Other random guessing:
- consistency ≈ 25%;
- max(accuracy) = 75%
  with 0 completeness;
- max(completeness) = 100%
  with 25% accuracy.
Conclusion

- Our model provides a scientific model of program complexity to understand and improve software process;

- Our theory of program complexity embodies the empirical research and extends formal models in this area;

- Our technique of measure evaluation demonstrates the usability of our theory in solving software engineering problems;

- Our model appears valid and effective as demonstrated by the multiple applications.